PURDUE UNIVERSITY

SCHOOL OF ELECTRICAL AND
COMPUTER ENGINEERING
WEST LAFAYETTE, IN 47907-1285 USA

W. KENT FUCHS
HEAD

August 21, 1998

Colin E. Wood
ONR 312
Office of Naval Research
Ballston Centre Tower One
800 North Quincy Street
Arlington, VA 22217-5660

Dear Dr. Wood:

Enclosed is a copy of the final version of the paper listed below concerning research supported in part by the Department of the Navy and managed by the Office of the Chief of Naval Research under Grant N00014-97-1-1013:

> V. Boppana and. K. Fuchs, "Dynamic Fault Collapsing and Diagnostic Test Pattern Generation for Sequential Circuits," *Proceedings of the IEEE International Conference on Computer-Aided Design*, to appear: Nov. 1998.

This material is covered under Distribution Statement A (Approved for Public Release: Distribution is Unlimited).

We appreciate your support of this research.

Sincerely,

W. Kent Fuchs
Head and Professor
Electrical and Computer Engineering

Enclosure: manuscript

cc: Administrative Grants Officer (Chicago IL)
Director, Naval Research Laboratory (Washington DC)
Defense Technical Information Center (Ft. Belvoir VA)

1285 Electrical Engineering Building
email: fuchs@purdue.edu

Phone: (765) 494-3539
Fax: (765) 494-3544

ICCAD 1998 (Nov.)
FINAL VERSION

DYNAMIC FAULT COLLAPSING AND DIAGNOSTIC TEST PATTERN GENERATION FOR SEQUENTIAL CIRCUITS

Vamsi Boppana
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway
Sunnyvale, CA 94086
vboppana@fla.fujitsu.com

W. Kent Fuchs
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
fuchs@purdue.edu

Abstract

In this paper, we present results for significantly improving the performance of sequential circuit diagnostic test pattern generation (DATPG). Our improvements are achieved by developing results that permit dynamic, fully functional, collapsing of candidate faults. Fault collapsing permits the organization of faults into disjoint partitions based on the indistinguishability relation. These results are used to develop a diagnostic test pattern generation algorithm that has the same order of complexity as that of detection-oriented test generation (ATPG). Techniques to identify untestable faults, based on exploiting indistinguishability identification, are also presented. Experimental results are presented on the ISCAS 89 benchmark circuits.

1 Introduction

The aim of diagnostic test generation is to provide test vectors that can distinguish between every distinguishable fault pair and to prove the remaining pairs of faults to be indistinguishable. This is in contrast with detection-oriented test generation which aims at generating test vectors that detect every detectable fault and proving the remaining faults to be undetectable. The terms *distinguishable*, *indistinguishable*, *detectable* and *undetectable* take on different meanings with different test methodologies (multiple observation time [1, 2], single observation time [1, 2] or conventional, gate-level test generation with single observation time and three-valued simulation [3]). However, irrespective of the test methodology, proving indistinguishabilities and undetectabilities is computationally intensive. Previous research on diagnostic test generation has primarily focused on combinational circuits [4–7]. Research on indistinguishability identification has also been devoted mainly toward combinational circuits [8–11]. The words equivalence and indistinguishability carry the same meaning for combinational circuits; however, such is not the case for sequential circuits, where the definition of sequential indistinguishability itself is dependent upon the test methodology used. Recent work [12, 13] has defined sequential indistinguishability under varying test strategies and simulation capabilities. That work derived results on avoiding the explicit proving of specific indistinguishability relations. Results on diagnostic test pattern generation for large sequential circuits have also been reported in recent work [14]. However, in contrast with this present paper, there was no attempt in the previous work to re-use the results already obtained during DATPG. The focus there was on developing an efficient diagnostic engine that, when given a pair of stuck-at faults, either generates a test sequence that distinguishes the two faults or concludes that they are indistinguishable.

In this paper, we develop results that permit the dynamic collapsing of faults for sequential circuits. Dynamic collapsing refers to the process of collapsing faults *during* diagnostic test pattern generation. This is in contrast with previous work on fault collapsing in sequential circuits [15] where the objective was to collapse faults statically (without a diagnostic test generator and possibly even before any automatic test pattern generation). Our fault collapsing results are used in organizing specific classes of faults into disjoint partitions based on the indistinguishability relation. These results are used to reduce the complexity of diagnostic test generation algorithm (DATPG) to the same order of complexity as that of detection-oriented test generation (ATPG). This significant reduction is achieved by the efficient reuse of the intermediate results provided by the DATPG algorithm.

2 Definitions

In this section, we briefly review the definitions of sequential indistinguishability under varying test strategies. A previous paper [12] developed these definitions for the context of fault diagnosis. The definitions are summarized here for clarity. The test strategies considered are the multiple observation time strategy (MOTS) [1, 2, 16], and the conventional gate level test generation strategy (using three-valued simulation to evaluate logic values) [3]. The conventional gate-level test generation strategy can in fact be considered to be a restricted form of the single observation time strategy (RSOTS), because of a possible loss in accuracy due to simulation. Concepts discussing various test

This research was supported in part by the Office of Naval Research (ONR) under grant N00014-97-1-1013, by the Defense Advanced Research Projects Agency (DARPA) under contract DABT 63-96-C-0069, and by the Semiconductor Research Corporation (SRC).

strategies for distinguishing sequential machines have also been suggested in the context of design verification [17, 18].

Sequential indistinguishability with MOTS

Definition 1 (Indistinguishability) *A fault pair (f_1, f_2) is said to be indistinguishable if there exist states S^{f_1} and S^{f_2} in the faulty machines corresponding to fault f_1 and f_2 respectively, and there exists* **no** *input sequence Y, such that the sequence $Z^{f_1}(Y, S^{f_1})$ produced by the faulty machine corresponding to f_1 is different from the sequence $Z^{f_2}(Y, S^{f_2})$ produced by the faulty machine corresponding to f_2.*

Sequential indistinguishability with RSOTS

As noted earlier, the restricted single observation time test generation strategy represents practical gate-level test generation with three-valued (0,1,X) logic simulation. The definition of indistinguishability, defined for the previous test generation strategy, has worked with the assumption that there is no loss of accuracy that **practical** gate-level test generation procedures, targeting large sequential circuits, suffer from due to the evaluation of unknown values. We now provide the following definition to characterize sequential indistinguishability in a practical gate-level test generation environment that uses three-valued logic simulation.

Definition 2 (Indistinguishability) *A fault pair (f_1, f_2) is said to be indistinguishable if there exists no input sequence Y such that the output responses produced by the two faulty machines, when starting from the fully unspecified state and as determined by three-valued logic simulation, are different on a* specific *time unit and on a specific primary output (i.e., 0 in one machine and 1 in the other or vice versa).*

3 Fault Collapsing and DATPG

Requirements

Dynamic fault collapsing proves valuable for diagnostic test generation because all future operations involved in diagnostic test generation, after a collapsing operation, need to consider only one of the two faults. Savings are achieved both during the test generation process as well as during diagnostic fault simulation (required to evaluate the test vectors generated during the DATPG process). Let us consider the indistinguishable fault pair (f_1, f_2). Our goal is to obtain conditions that enable the collapsing of these two faults into a single fault and not affect the rest of the diagnostic test generation process. It is worth noting that the conditions that we actually check for collapsing also ensure that the faulty machines in question produce "identical" responses.

Definition 3 (Fault Collapsing Requirements) *A fault pair (f_1, f_2) can be collapsed if for any arbitrary fault f_3 different from both faults f_1 and f_2 the following two conditions are satisfied.*
Condition 1. *(f_1, f_3) identified as indistinguishable implies (f_2, f_3) indistinguishable and vice versa.*
Condition 2. *(f_1, f_3) distinguished implies (f_2, f_3) distinguished and vice versa.*

We first note the dynamic fault collapsing result for combinational circuits. Let us consider two faults f_1 and f_2.

Observation: If the pair of faults (f_1, f_2) is declared to be equivalent (indistinguishable) during diagnostic test generation in a combinational circuit, then they may be collapsed into a single class of faults and may be represented by either one of the faults.

Example

An example illustrating the use of dynamic fault collapsing is presented in Figure 1. In the example, two DATPG algorithms are traced to illustrate fault collapsing. The first algorithm runs without any fault collapsing, and the second algorithm runs with fault collapsing. The column **Current Status** represents a set of classes of faults. Faults from distinct classes have already been distinguished from each other, and faults in the same class have not yet been distinguished. The column **Candidate** shows the current fault pair input to the DATPG algorithm. The column **Result** gives one of three results **I, D** or **A**, standing for indistinguishable, distinguished or aborted, respectively. The last column **Proven** carries different meanings for the two algorithms. For the case without any fault collapsing, it contains fault pairs proven to be indistinguishable from each other, while with fault collapsing, the entries are sets of classes of faults. Each class represents faults that are proven to be indistinguishable from each other.

We can see from the traces of this example that savings are achievable in the number of indistinguishability relations explicitly proven by the DATPG algorithm. The DATPG algorithm is simplified by retaining only one of the faults for each time an indistinguishability relation is identified. For example, we note that fault 2 is eliminated from the current status after Step 1 because fault 1 represents it for all future operations. The example shows that the number of indistinguishability relations that need to be proven explicitly by a call to the core diagnostic test generation routine has been reduced from 9 to 5. Fault pairs (2 3), (2 4), (3 4) and (6 7) have been proven as a result of the collapsing operations.

It is also useful to note the reductions provided in the fault simulation operations while evaluating the test vectors obtained when a fault pair is distinguished. As an example, the diagnostic fault simulator used to evaluate the capability of the vectors provided after Step 4 need not perform any simulation or evaluation operations for faults 2, 3 and 4. The collapsed list of faults is automatically available

DATPG algorithm trace *without* collapsing

Step	Current Status	Candidate	Result	Proven
1	(1 2 3 4 5 6 7 8)	(1 2)	I	[1 2]
2	(1 2 3 4 5 6 7 8)	(1 3)	I	[1 2] [1 3]
3	(1 2 3 4 5 6 7 8)	(1 4)	I	[1 2] [1 3] [1 4]
4	(1 2 3 4 5 6 7 8)	(1 5)	D	[1 2] [1 3] [1 4]
5	(1 2 3 4) (5 6 7) (8)	(2 3)	I	[1 2] [1 3] [1 4] [2 3]
6	(1 2 3 4) (5 6 7) (8)	(2 4)	I	[1 2] [1 3] [1 4] [2 3] [2 4]
7	(1 2 3 4) (5 6 7) (8)	(3 4)	I	[1 2] [1 3] [1 4] [2 3] [2 4] [3 4]
8	(1 2 3 4) (5 6 7) (8)	(5 6)	I	[1 2] [1 3] [1 4] [2 3] [2 4] [3 4] [5 6]
9	(1 2 3 4) (5 6 7) (8)	(5 7)	I	[1 2] [1 3] [1 4] [2 3] [2 4] [3 4] [5 6] [5 7]
10	(1 2 3 4) (5 6 7) (8)	(6 7)	I	[1 2] [1 3] [1 4] [2 3] [2 4] [3 4] [5 6] [5 7] [6 7]
11	(1 2 3 4) (5 6 7) (8)	DONE	-	[1 2] [1 3] [1 4] [2 3] [2 4] [3 4] [5 6] [5 7] [6 7]

† Number of indistinguishability relations explicitly proven by DATPG = 9

DATPG algorithm trace *with* collapsing

Step	Current Status	Candidate	Result	Proven
1	(1 2 3 4 5 6 7 8)	(1 2)	I	[1 2]
2	(1 3 4 5 6 7 8)	(1 3)	I	[1 2 3]
3	(1 4 5 6 7 8)	(1 4)	I	[1 2 3 4]
4	(1 5 6 7 8)	(1 5)	D	[1 2 3 4]
5	(1) (5 6 7) (8)	(5 6)	I	[1 2 3 4] [5 6]
6	(1) (5 7) (8)	(5 7)	I	[1 2 3 4] [5 6 7]
7	(1) (5) (8)	DONE	-	[1 2 3 4] [5 6 7]

† Number of indistinguishability relations explicitly proven by DATPG = 5
†† Savings are achieved in fault simulation after a distinguished (D) result from DATPG

Figure 1: Example illustrating the savings achievable with dynamic fault collapsing

as the set of classes in the column **Proven** at the end of the procedure. Fault collapsing cannot be applied to sequential circuits based on indistinguishability identification alone because of a possible failure in satisfying one of the two identified conditions. However, we show in the next section that it is possible to identify conditions that may be checked easily and permit the collapsing operations for sequential circuits under the two test strategies considered. The conditions developed are based on synchronizing sequences and strong connectivity (for MOTS) and based on initializing sequences (for RSOTS). It is also reiterated that we do not intend in anyway to find these sequences, but provide techniques to make use of knowledge concerning the existence of these sequences based on either existing (possibly detection-oriented) test vectors or information provided by designers.

3.1 Fault collapsing

We shall now identify conditions for fault collapsing under the restricted single observation time test strategy (RSOTS).

Definition 4 (Initializability) *A machine M is initializable with three-valued logic simulation if there exists an input sequence Y, such that the resulting state of M (evaluated by three-valued simulation) is fully specified on the application of Y, when the initial state is fully unspecified (consisting of all Xs and corresponding to the entire state space). Initializability is thus synchronizability subject to three-valued logic simulation.*

Theorem 1 *Consider two faults f_1 and f_2. Let the machines corresponding to these two faults be M^{f_1} and M^{f_2} respectively. If the fault pair (f_1, f_2) is identified to be indistinguishable and if the machines M^{f_1} and M^{f_2} are* **initializable**, *then the fault pair (f_1, f_2) can be collapsed and either one of the faults f_1 or f_2 can be used to represent the collapsed class of faults.*

Proof We shall demonstrate that both the conditions that are required to be proven for fault collapsing are satisfied. Consider any arbitrary fault f_3 different from the faults f_1 and f_2.

Condition 1. Let (f_1, f_3) be indistinguishable. We now demonstrate (by contradiction) that the fault pair (f_2, f_3) is also indistinguishable. Let us assume that the fault pair (f_2, f_3) is distinguishable. This implies that there exists an input sequence Y, such that the output responses produced by the two faulty machines, when starting from the fully unspecified state and as determined by three-valued logic simulation, are different on a specific time unit t and specific primary output o (i.e., 0 in one machine and 1 in the other or vice versa). Without loss of generality, assume that the value produced by M^{f_2} is 0 and the value produced by M^{f_3} is 1. Consider the output response produced by M^{f_1}, starting from the fully unspecified state and as determined by three-valued simulation, to the input sequence Y. This output response must assume a value of X at time t on primary output o (otherwise, one of the pairs (f_1, f_2) or (f_1, f_3) becomes distinguishable). However, it is known

that the machine M^{f_1} is initializable with three-valued simulation. Let P be an input sequence that initializes machine M^{f_1}. Consider the application of the input sequence created by concatenating Y at the end of P to the machine M^{f_1} starting from the fully unspecified state. The output response of the machine, as evaluated by three-valued simulation, would distinguish either (f_1, f_2) or (f_1, f_3), at the same vector and primary output that distinguished (f_2, f_3). Hence, we arrive at a contradiction. Because f_1 and f_2 can be used interchangeably in the above argument, we have the first condition.

Condition 2. Let (f_1, f_3) be distinguished by the test vector sequence T_D. We now demonstrate that the fault pair (f_2, f_3) is also distinguished. Because the pair of faults (f_1, f_3) is distinguished by T_D, there is a primary output on which the responses for f_1 and f_3 are $0(1)$ and $1(0)$, respectively. Without loss of generality, assume that the value produced by M^{f_1} is 0 and the value produced by M^{f_3} is 1. Consider the response produced by M^{f_2} at the exact same primary output to the same sequence of vectors that produced the difference in values between f_1 and f_3. The output response could only be either the value X or the value 0. Now, because there is an initializing sequence of vectors for the machine M^{f_2}, say Q, the response of the machine M^{f_2} to the concatenated sequence Q followed by T_D distinguishes the pair (f_2, f_3). Again, because f_1 and f_2 have been used interchangeably in the above argument, we have the second condition.

The following result, derived from the above result, exploits the work done by detection-oriented test generation under the conventional test generation strategy.

Corollary 1 *Any faulty machine proven to be undetectable can be collapsed with the good machine provided the fault-free and the faulty machines are initializable.*

3.1.1 Untestability Identification based on Indistinguishability Identification

We now present two related results that can be used for the rapid proving of undetectability by the use of sequential indistinguishability identification. The results provide a method to prove a fault untestable by pairing it with an already proven untestable fault that is initializable and concluding that the fault pair is indistinguishable. If the fault that is proven to be untestable is combinationally redundant, then it may be possible to make a stronger claim regarding the kind of untestability that can be identified.

Corollary 2 (Untestability identification) *Consider a fault f_1, aborted by detection-oriented test generation. Let f_2 be proven undetectable. Then, if the pair of faults (f_1, f_2) is proven to be indistinguishable and if the faulty machine corresponding to f_2 is initializable, then the fault f_1 is undetectable.*

Corollary 3 (Untestability identification) *Consider a fault f_1, aborted by detection-oriented test generation. Let f_2 be proven combinationally redundant and initializable. Then, if the pair of faults (f_1, f_2) is proven to be indistinguishable, the fault f_1 is undetectable.*

Fault collapsing results similar to the one in Theorem 1 have also been obtained for the multiple observation time test strategy (MOTS). These results are based on utilizing existing knowledge of synchronizing sequences and connectivity information [19, 20] in the state transition graph of the circuit. These results are not presented here due to lack of space, but are available in a technical report [13].

3.2 Creation of mathematical equivalence classes

The properties of sequential fault collapsing developed are further exploited to show that they help organize the set of faults that satisfy specific properties (connectivity, synchronizability or initializability) into disjoint partitions. This is a key property that provides a remarkable reduction in the complexity of diagnostic test pattern generation.

The term *mathematical equivalence class* is used here as opposed to just *equivalence class* in order to make a distinction between the formal definition of equivalence as defined in the domain of set theory and the definition from the digital circuit domain.

Now, let us take the set of the faults known to be initializable and consider the indistinguishability relation on this set of faults. We now show that this relation is indeed a mathematical equivalence relation.

Theorem 2 (Init'ble + Indist'ble $\Rightarrow$ Disjoint Partition) *Let the set of initializable faults be denoted by I and the relation R under consideration be the indistinguishability relation. Then,*

- $(a, a) \in R$ *(reflexive)*
- $(a, b) \in R$ *implies that* $(b, a) \in R$ *(symmetric)*
- $(a, b) \in R$ *and* $(b, c) \in R$ *implies that* $(a, c) \in R$ *(transitive; from our indistinguishability results)*

We exploit the well-known property of equivalence relations which states that the equivalence classes are all disjoint. Results similar to the above result are also achieved for the multiple observation time test strategy by replacing the initializability property with the appropriate properties of connectivity and synchronizability. These results are not presented here for lack of space.

4 DATPG

In this section, we present the new DATPG algorithm equipped to exploit the dynamic collapsing of faults and the implicit identification of indistinguishability. The purpose of the DATPG algorithm is to generate test sequences

that distinguish every distinguishable fault pair and prove the rest indistinguishable. Of course, if the algorithm is unable to either generate a test or prove the fault pair to be indistinguishable, then it simply aborts on that fault pair. The DATPG algorithm is typically interfaced with diagnostic fault simulation to identify any additional fault pairs that may have been distinguished whenever a targeted fault pair is distinguished.

```
// Routine GetNextFaultPair() gets a new fault pair into
// f1 and f2; if there are no more pairs, it returns NULL
// Routine Datpg(f1, f2) returns the status of the DATPG operation as
// DISTINGUISHED, INDISTINGUISHABLE OR ABORTED
// If the status is DISTINGUISHED, the test sequence obtained is denoted by V
// Routine ClassSplitDropPairs(V) performs fault simulation to determine additional
// pairs distinguished by appending the test sequence V to the existing set
// of test vectors and to split the existing classes into smaller ones
// Routine MarkIndistinguishable(f1, f2) marks the pair indistinguishable
// Routine CollapsingConditionsSatisfied(f1, f2) checks if the conditions
// identified for the test generation paradigm being used are satisfied.
// Routine Collapse(f1, f2) chooses one of the faults to
// represent both the classes for the rest of the DATPG algorithm
// and drops the other fault for future operations
// Routine IdentifyImplicitIndistinguishabilities(f1, f2) checks to identify
// additional indistinguishabilities proven based on the currently proven relation
// Routine MarkAborted(f1, f2) marks the fault pair as aborted

while ((f1, f2) = GetNextFaultPair()) {
    if(Datpg(f1, f2) == DISTINGUISHED))
        ClassSplitDropPairs(V);
    elseif (Datpg(f1, f2) == INDISTINGUISHABLE)) {
        if (CollapsingConditionsSatisfied(f1, f2)) {
            Collapse(f1, f2); // Dynamic fault collapsing here
        }
        else {
            MarkIndistinguishable(f1, f2);
            IdentifyImplicitIndistinguishabilities(f1, f2);
            // Implicit Identification
        }
    }
    else { // ABORTED
        MarkAborted(f1, f2);
    }
}
```

Figure 2: New DATPG algorithm

The new DATPG algorithm, is shown in Figure 2. We note again that it is necessary to store a fault pair explicitly as indistinguishable in the new algorithm only when the conditions required for collapsing the pair into a single fault fail. (This occurs only for sequential circuits and occurs precisely when initialization cannot be verified for at least one of the faults.) We also note that the sequentially collapsed list of faults is available at the completion of the new algorithm.

4.1 On the complexity of DATPG

In this subsection, we show that the complexity of diagnostic test pattern generation, using fault collapsing and implicit identification of sequential indistinguishability, is of the same order of complexity as detection-oriented test pattern generation. In order to estimate the computational complexity of the test generation algorithms, a basic operation that can be used to measure the number of steps required by a test generation algorithm is introduced. This operation is illustrated in Figure 3. A basic diagnostic engine is invoked either to distinguish a pair of faults or to claim that the pair is indistinguishable. It is to be noted that both diagnostic test generation and detection-oriented test generation can be modeled in terms of this basic framework, with the pair passed to the diagnostic engine consisting of the good and the faulty machine in the case of detection-oriented test generation and consisting of two faulty machines for diagnostic test generation. It is also pointed out that, in this analysis, the complexity of fault simulation is not considered because simulation is typically significantly less expensive than test pattern generation.

Let us now assume that each query to the diagnostic engine involves a cost of one computational unit. This, of course, is not necessarily a correct assumption but it is instructive to analyze the complexity of both the procedures subject to this assumption. Let the total number of faults be f and the number of initializable faults be f_i. Let us also assume that the fraction of total faults that are initializable is p. Further, let us assume that $q = 1 - p$. We will demonstrate later that the typical values of p are close to 1 through experimental results. We first note that using the simple DATPG algorithm where each pair of faults is explicitly passed to the DATPG engine, the complexity of the entire procedure becomes $f(f-1)/2$. We now demonstrate that our improved results permit the reduction of this complexity to $O(f)$ for typical values of p.

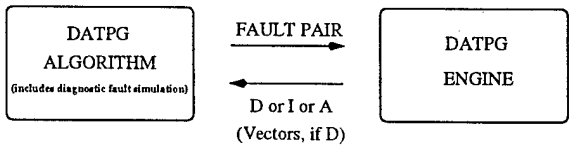

Figure 3: DATPG and the diagnostic engine

Let us first finish the DATPG operation on pairs produced only by the set of initializable faults f_i. The rest of the pairs will be considered later to complete the DATPG algorithm. At the start of the DATPG algorithm, all the faults trivially belong to the same class. First, let us assume that the diagnostic engine being used in the model is perfect, i.e., the answer provided by the diagnostic engine is either distinguished or proven to be indistinguishable. We shall consider the case of aborted faults later in the analysis. At each step of the DATPG algorithm, a fault pair whose status is as yet unknown (i.e., both the faults belong to the same current class) is passed to the diagnostic engine. The result provided by the diagnostic engine introduces the following changes in the current list of classes:

- If the result of the operation is distinguished, then the

number of classes must increase by at least 1. (Of course, the increase may be much larger because the vectors produced may incidentally partition the classes into several classes.)

- If the result of the operation is indistinguishable, then the number of faults decreases by 1 because the two faults can now be collapsed into a single fault.

However, the maximum number of classes possible is limited by f_i and the minimum number of faults remaining in the DATPG algorithm is limited by 1. It is also clear that both the numbers (number of classes and number of faults) vary monotonically. Hence, the maximum number of steps required by the DATPG algorithm is limited by $2f_i$, i.e., $2pf$. The number of remaining fault pairs is given by $pqf^2 + qf(qf-1)/2$ which is less than $pqf^2 + q^2f^2/2$. Therefore, the total cost of the diagnostic procedure can be represented by $2pf + pqf^2 + q^2f^2/2$. If q is sufficiently small, then this expression has the complexity $O(f)$.

Now, if we remove the assumption about aborted faults and assume that the diagnostic engine does abort fault pairs, then the analysis can be modified appropriately. If the total number of fault pairs aborted by the diagnostic engine is a, then that is exactly the additional number of operations that may be required by the DATPG algorithm (because an aborted fault pair neither permits a collapsing operation nor contributes to an increase in the number of classes and is, in that sense, a wasted computation). The cost of the DATPG algorithm is thus represented by the expression $2pf + pqf^2 + q^2f^2/2 + a$. If the number of aborted fault pairs can be assumed to be small (by a good diagnostic engine), then the complexity can still be seen to be $O(f)$.

It is easy to see that even a detection-oriented ATPG algorithm can indeed take $O(f)$ steps (if each faulty machine is initialized, the corresponding fault is distinguishable from each of the other faults and there are no incidental detections). Hence, this analysis shows that both DATPG and ATPG are of the same order of complexity.

We also note that the above analysis can be repeated for the multiple observation time test strategy assuming either the property of strong connectivity or synchronizability (similar to the use of initializability here) to arrive at the same conclusions about the complexity of DATPG.

Having performed the above analysis on the complexities of DATPG and ATPG, we now present a caveat that is introduced by the practical use of these procedures. In practice, DATPG often follows ATPG; this implies that many of the fault pairs that are *easy to distinguish* are distinguished incidentally. Hence, the fault pairs that are left to be targeted explicitly by the DATPG algorithm are often *difficult* and hence may contribute to the additional difficulty of the DATPG procedure. In addition, the assumption about the cost of the simulation step being much less than that of test generation step may not always be correct.

5 Experimental Results

Results on initialization and DATPG

In this section, we demonstrate the improvements provided by the results developed in this paper using the restricted single observation time test strategy. This is the chosen strategy because of its practicality with respect to large circuits and its wide-spread use. Because the test strategy is RSOTS, the state property of concern for enabling the results developed is initializability. As we have already noted, a machine corresponding to an *initialized* fault is *initializable*. Because it is easy to verify the initialization of a faulty machine with a *given* set of vectors, that is our approach.

Results that show the improved computations in DATPG possible using the improved DATPG algorithm are presented in Table 1. Recall that improvements in complexity were achieved from the original value of $f(f-1)/2$ to the improved value of $2pf + pqf^2 + qf(qf-1)/2$, where p is the fraction of initializable faults and q is the fraction of faults not initialized. The results presented in the table show the total faults, the number of initialized faults (using HITEC/STG3 [21,22] vectors), the number of faults that are not initialized, the number of pairs that would have to be handled by the old DATPG algorithm, the number of pairs that would have to be handled by the new DATPG algorithm and the percentage ratio between the new and the old costs. These results show that it is indeed possible to tackle the DATPG problem for the *entire* set of faults for many practical circuits.

Results on Fault Collapsing

The results of a fault collapsing experiment are shown in Table 2. The first column of the table shows the benchmark circuit and the second column shows the total number of faults. These faults were processed to first remove combinationally (full-scan) redundant faults (to obtain the number Nfsr). The obtained faults are then collapsed by applying a combinational equivalence prover (DIATEST [6]). The resulting number of faults is shown by the column FscNfsr. Undetected faults (using HITEC [21] vectors) are then removed from this list of faults to obtain the set of faults, indicated by NuFscNfsr. These faults are processed with the new fault collapsing diagnostic test generator that was built on top of the diagnostic test generator DIAGGEN [14] to obtain the sequentially collapsed list of faults, SeqcNuFscNfsr. The untestable faults that were removed were then added back to this list of faults to get the list of faults indicated by SeqcFscNfsr. The final list of sequentially collapsed faults is obtained by adding back representative faults from the combinationally redundant faults (just one fault is sufficient when the good circuit is initializable) and is shown by column SeqcFsc. It is clear from the numbers

Table 1: DATPG comparison

Ckt.	Total Faults	Init.	Not Init.	Old DATPG #Pairs	New DATPG #Pairs	% Ratio (New/Old)
s298	308	299	9	47278	3325	7.03
s344	342	329	13	58311	5013	8.60
s400	428	413	15	91378	7126	7.80
s526	555	538	17	153735	10358	6.74
s641	467	460	7	108811	4161	3.82
s713	581	574	7	168490	5187	3.08
s820	850	849	1	360825	2547	0.71
s832	870	869	1	378015	2607	0.69
s953	1079	3	1076	581581	581584	100.00
s1238	1355	1355	0	917335	2710	0.30
s1423	1515	1450	65	1146855	99230	8.65
s1488	1486	1484	2	1103355	5937	0.54
s1494	1506	1504	2	1133265	6017	0.53
s5378	4603	4567	36	10591503	174176	1.64
s35932	39094	39084	10	764150871	469053	0.06

Table 2: Fault collapsing results

Ckt.	Faults	Nfsr	FscNfsr	NuFscNfsr	SeqcNuFscNfsr	SeqcFscNfsr	SeqcFsc
s298	308	308	288	256	183	215	216
s344	342	342	337	330	255	262	263
s526	555	554	523	128	102	497	498
s641	467	467	460	405	337	392	393
s713	581	542	469	413	338	394	395
s820	850	850	814	778	755	791	792
s832	870	856	816	777	755	794	795
s1238	1355	1286	1251	1248	1209	1212	1213
s1423	1515	1501	1361	553	430	1238	1239
s1488	1486	1486	1465	1426	1395	1434	1435
s1494	1506	1494	1469	1431	1401	1439	1440
s5378	4603	4563	4190	2921	2645	3914	3915
s35932	39094	35110	25476	25235	25235	25476	25477

in this table that the size of the fault lists may be considerably reduced beyond the standard structural fault collapsing [3] that is widely used.

Results of Untestability Identification using Indistinguishability Identification

In the experiments performed here, the set of aborted faults produced by the HITEC test generator [21] is taken and pairs of faults are created by pairing them with (initializable) combinationally redundant faults. These fault pairs are injected in the last (rightmost) frame of a combinationally expanded version of the sequential circuit (expanded 10 time frames in this experiment) and their equivalence is examined (by the diagnostic test generator DIATEST [6]). If the fault pair is proven to be equivalent, then the fault is proven to be untestable in the original, sequential, circuit.

Table 3 provides the untestability identification results. The number of initializable, combinationally redundant faults is provided in column 4. Results are presented only for those circuits for which at least one initializable, combinationally untestable fault could be identified. Columns 1, 2, 3, 5, 6 and 7 stand for circuit name, number of proven untestable faults, number of aborted faults, number of candidate pairs, number of pairs proven to be indistinguishable and the number of faults proven to be untestable, respectively. The last column presents the CPU time in seconds, as measured on a SUN Ultrasparc1 workstation with 58 MB of main memory. These results demonstrate that it is possible to utilize the power of indistinguishability identification even for untestability identification.

Table 3: Untestability identification

Ckt.	Untest. Faults	Aborted Faults	Comb. Red. Faults	Candidate Pairs	Proven Pairs	Proven Faults	Time (sec)
s526	17	487	1	487	21	21	1190
s832	46	8	14	112	70	5	234
s1423	11	949	14	13286	32	5	14433
s1494	40	17	12	204	132	11	1066
s5378	148	1303	40	1303	618	618	17479
s35932	3984	10	3984	1000	0	0	123

6 Summary

State space properties were used to derive conditions under which sequential fault collapsing can be permitted. This was used to provide a result that partitions the set of faults into disjoint classes. This partition is arrived at by demonstrating that the class structure corresponds to a mathematical equivalence class structure. The results presented permit the design of a diagnostic test generation algorithm that has the same order of complexity as a regular, detection-oriented test generation algorithm. The results have also been effectively used for identifying untestable faults. Experiments on benchmark circuits were used to demonstrate the results.

References

[1] I. Pomeranz and S. M. Reddy, "The multiple observation time test strategy," *IEEE Trans. Computer-Aided Design*, vol. 40, no. 5, pp. 627–637, May 1992.

[2] I. Pomeranz and S. M. Reddy, "Classification of faults in synchronous sequential circuits," *IEEE Trans. Computers*, vol. 42, no. 9, pp. 1066–1077, Sept. 1993.

[3] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital System Testing and Testable Design*, New York, NY: Computer Science Press, 1990.

[4] J. Savir and J. P. Roth, "Testing for, and distinguishing between failures," in *Proc. Intl. Symp. Fault Tolerant Computing*, June 1982, pp. 165–172.

[5] P. Camurati, D. Medina, P. Prinetto, and M. Sonza Reorda, "A diagnostic test pattern generation algorithm," in *Proc. Intl. Test Conf.*, Sept. 1990, pp. 52–58.

[6] T. Grüning, U. Mahlstedt, and H. Koopmeiners, "DIATEST: A fast diagnostic test pattern generator for combinational circuits," in *Proc. Intl. Conf. Computer-Aided Design*, Nov. 1991, pp. 194–197.

[7] I. Hartanto, V. Boppana, and W. K. Fuchs, "Diagnostic fault equivalence identification using redundancy information & structural analysis," in *Proc. Intl. Test Conf.*, Oct. 1996, pp. 294–302.

[8] E. J. McCluskey and F. W. Clegg, "Fault equivalence in combinational logic networks," *IEEE Trans. Computers*, vol. C-20, no. 11, pp. 1286–1293, Nov. 1971.

[9] A. Goundan and J. P. Hayes, "Identification of equivalent faults in logic networks," *IEEE Trans. Computers*, vol. C-29, no. 11, pp. 978–985, Nov. 1980.

[10] B. K. Roy, "Diagnosis and fault equivalences in combinational circuits," *IEEE Trans. Computers*, vol. C-23, no. 9, pp. 955–963, Sept. 1974.

[11] A. Lioy, "Advanced fault collapsing," *IEEE Design & Test of Computers*, vol. 9, no. 1, pp. 64–71, Mar. 1992.

[12] V. Boppana, I. Hartanto, and W. K. Fuchs, "Characterization and implicit identification of sequential indistinguishability," in *Proc. Intl. Conf. VLSI Design*, Jan. 1997, pp. 376–380.

[13] V. Boppana, "State information-based solutions for sequential circuit diagnosis and testing," Ph.D. thesis, Center for Reliable and High-Performance Computing, University of Illinois at Urbana-Champaign, Tech. Rep. CRHC-97-20, July 1997.

[14] I. Hartanto, V. Boppana, J. H. Patel, and W. K. Fuchs, "Diagnostic test pattern generation for sequential circuits," in *Proc. VLSI Test Symp.*, Apr. 1997, pp. 196–202.

[15] J. E. Chen, C. L. Lee, and W. J. Shen, "Single Fault Fault Collapsing Analysis in Sequential Logic Circuits," *IEEE Trans. Computer-Aided Design*, vol. 10, no. 10, pp. 1559–1568, Oct. 1991.

[16] I. Pomeranz and S. M. Reddy, "On improving fault diagnosis for synchronous sequential circuits," in *Proc. Design Automation Conf.*, June 1994, pp. 504–509.

[17] C. Pixley, "A theory and implementation of sequential hardware equivalence," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 12, pp. 1469–1494, Dec. 1992.

[18] S. Y. Huang, K. T. Cheng, K. C. Chen, and U. Glaeser, "An ATPG-based framework for verifying sequential equivalence," in *Proc. Intl. Test Conf.*, Oct. 1996, pp. 865–874.

[19] Z. Kohavi, *Switching and Finite Automata Theory*, New York, NY: McGraw-Hill, 1978.

[20] F. C. Hennie, *Finite-State Models for Logical Machines*, New York, NY: John Wiley & Sons, Inc., 1968.

[21] T. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," in *Proc. European Design Automation Conf.*, Feb. 1991, pp. 214–218.

[22] W. T. Cheng and S. Davidson, "Sequential circuit test generator STG benchmark results," *IEEE Intl. Symp. Circuits and Systems*, pp. 1938–1941, May 1989.